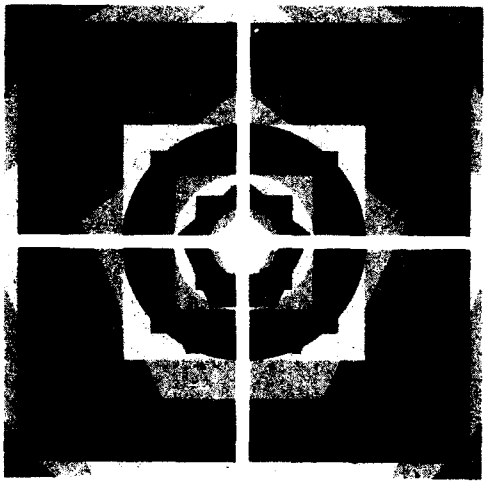
M92-B0000009

(1)

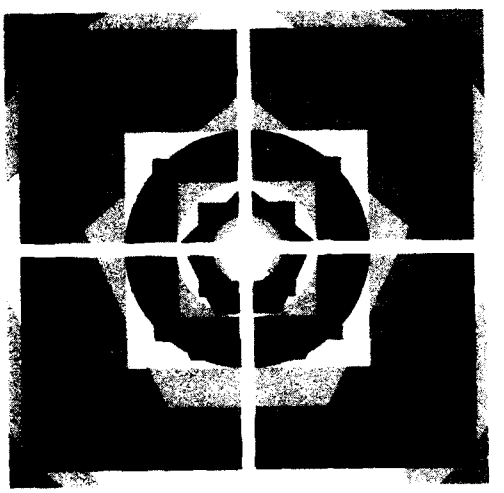
Development of DOD C³I Systems Using Commercial Off-the-Shelf Products



Dr. Barry M. Horowitz

M92-B0000009

Development of DOD C^{3}I Systems Using Commercial Off-the-Shelf Products



Dr. Barry M. Horowitz

January 1992

The MITRE Corporation
Burlington Road
Bedford, MA 01730

PREFACE

This document contains the text of a keynote address originally given by Dr. Barry M. Horowitz, President and Chief Executive Officer of The MITRE Corporation, at the MITRE-Bedford Software Center seminar *Using Off-the-Shelf Software in C³ Systems*.

ACKNOWLEDGMENTS

Many people at MITRE contributed data and ideas to this document. Special thanks are expressed to Judith A. Clapp and Roy W. Jacobus.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The use of commercial off-the-shelf (COTS) products in the development of command, control, communications, and intelligence (C³I) systems for the Department of Defense (DOD) is crucial. Significant reduction in the DOD budget and the expectation of further reductions will force the defense community to build lower cost systems that do not depend heavily on customized hardware and software. The concept of "open system architecture," where individual subsystems designed and manufactured by different industrial organizations can be integrated into systems of high performance (by virtue of common interface standards), is making its appearance in the commercial marketplace. This creates the possibility that large-scale military systems can be integrated at low cost from commercial products, with relatively short development schedules. The DOD will need to exploit this possibility.

An additional impetus for the use of commercial products is the public's (and military commanders') growing familiarity with products readily available on the market for home and business use. As part of their personal lives, military decision-makers are becoming aware of the powerful computing and software options that can be obtained off-the-shelf, and will be unwilling to believe that custom development is necessary to achieve desired performance. As a result, it is realistic to predict that commercial systems will become the norm for military C³I systems, with custom-designed equipment the exception.

Over the last few years, substantial effort has been spent in initial attempts to integrate commercially based products into military C³I systems. From these efforts, the defense community has developed a style and approach to COTS integration. Rapid prototyping is being employed extensively as part of the design process. Many professionals in the prototype arena are knowledgeable about open system standards; this was not true as recently as four or five years ago. Much more attention is being given to commercial standards, because they are becoming useful, and because organizations in the commercial world have been formed to implement them.

PARTICIPANTS AND THEIR ROLES

There are three major participants in a typical military development: (1) the using command, or user, who sets the system requirements and who has final responsibility for operating the system; (2) the development command, or developer, who is given responsibility for developing a system meeting the user's requirements; and (3) industry, who actually designs, produces, and tests both the development versions and the final production versions of the desired system. However, when the developments are software-intensive systems with an emphasis on the use of COTS products, these typical roles are not always followed. Several examples are illustrated in the following list of projects, for which the Electronic Systems Division (ESD) of the Air Force was the development command, supported by The MITRE Corporation. The development approach, and the roles apportioned to the three participants, were somewhat different in each case (see table 1).

- Sentinel Byte's mission is to make intelligence data originating from higher levels of command available to the pilot doing mission planning.
- Granite Sentry is a command post, located in the Cheyenne Mountain Complex in Colorado, for assessing ballistic missile warnings from our strategic radar and infrared sensors.
- NORTIC is a command post designed to help NORAD track drug smugglers.
- The Message Handling System receives messages on a network and disseminates them on the basis of full profiling.
- The Airborne Command and Control Center (ABCCC) is an airborne platform that carries operators and a large variety of radios for communicating during a crisis.
- The Mission Planning System is an automated aid to assist in the planning of air missions.

Table 1. Examples of ESD as Development Command on Software-Intensive Systems

System	User	User's Role	Developer's Role	Industry's Role
Sentinel Byte	USAFE, PACAF, TAC, MAC, SAC	Generated requirements	Built two COTS prototypes installed at AF sites. Success led to decision to buy 15 more	Replicated developer's prototypes
Granite Sentry	NORAD	Hired and managed support contractors to develop system	Generated and analyzed system architecture, maximizing COTS use	Provided software support to user
NORTIC	NORAD	Generated requirements	Developed system architecture and system design. Selected a contractor to buy COTS components. Selected another support contractor to integrate, test, and maintain system	Bought COTS components to implement developer's architecture. Integrated, tested, and maintained system
Message Handling System	DIA	Generated requirements	Generated specification. Hired contractor to demonstrate initial prototype	Designed and built prototype, maximizing COTS use
ABCCC	TAC	Generated requirements	Generated product specification. Selected two contractors to build prototypes	Designed and built prototypes and production version
Mission Planning System	TAC	Built 3/4 of system, asked developer to complete	Transitioned system to more open architecture, so that capability could grow by addition of COTS products	Designed and built system, using developer's architecture

JUDGMENT

The problem with using commercial products is relating the commercially available equipment and software to the mission needs. This problem involves the different skills and judgments listed in table 2.

Table 2. Issues and Judgments Regarding Commercially Available Equipment and Software

Issue	Kind of Judgment
What products are available on the market?	Oversight of entire commercial market
Can commercial products be integrated, even if individual products have adequate capability?	Technical judgment
Will the completed COTS system be operationally useful?	User judgment
How much will it cost to add capability later?	Technical judgment
How difficult will the COTS system be to modify?	Technical judgment
To what extent is the user willing to compromise on the system requirements, balancing what is desired versus what is readily attainable?	Mission judgment

IMPORTANT PHILOSOPHICAL ISSUES

In each of the projects described, different judgments were made about the mixture of roles which, when coupled with business decisions, led to different acquisition styles.

While there is no best acquisition style, the different styles used in the projects described above are not a sign of healthy variety. They did not result from a set of logical decisions, but instead were driven by the varying political and economic environments surrounding each development. No clear method or policy is available to aid in the selection of acquisition approaches. The defense community has not yet dealt adequately with some important philosophical issues regarding developments incorporating commercial products. Two of these issues relate to performance specifications and system architecture.

Performance Specifications

The problem of performance specifications in cases where COTS use is desired has not yet been satisfactorily resolved. Air Force system developments are traditionally begun by writing performance and functional specifications. These specifications explain what the government wants a contractor to do, thereby providing the contractor with a firm basis on which to accept the risk of building the hardware and software for the system at an estimated cost.

However, when a system is to be developed using commercial products, the design latitude available when compared to building a system of custom components is severely constrained. Design freedom is implicit in the idea of a specification. For programs driven by the desire to integrate unmodified commercial hardware and software products, freedom of design is a false assumption. The concept of writing a specification does not fit with the idea of building systems from commercial parts.

In principle, a conflict between specifications and the use of COTS products is not necessary. Before making a bid, industry could perform the trade-off studies needed to identify applicable COTS products, select those that represent the best match to the specification, and verify that the selected products could actually be integrated within the proposed schedule and cost. However, projects based on commercial products virtually always have small budgets, and it is unreasonable to expect a company to invest in extensive trade-off studies before it bids on a project that offers only a modest return.

In practice, industry must use its knowledge about commercial components (which currently is limited) to make a calculated guess about whether those products can be integrated to meet the specifications. A wrong guess results in no options for recovery, because typically no money has been budgeted by the government for any custom design. For example, some development projects at ESD have been cancelled because the efforts were started with a specification and were later discovered to be infeasible using the selected COTS products; both industry and the government took losses. A way must be found to remove specifications from the process of buying COTS-based systems.

System Architecture

A second problem not adequately resolved is system architecture, which affects life-cycle costs in software-intensive systems. A large percentage of a software system is changed over its life cycle, and a large fraction of the cost of owning a software system occurs after development. Various estimates indicate about 60 percent of the cost of software is incurred after it is shipped to the user. Of that fraction, about 70 percent is spent adding new features, either because the original capability was unsatisfactory or the mission changed. Similarly, at ESD, about half the software effort is spent reworking what was initially developed, even during development. In total, about 50 percent of C^3I software development efforts are spent responding to changes in the original specifications. The nature of C^3I development precludes holding firm to specifications over a long time, due to the rapid tempo of new commercial developments.

To accommodate such variability in requirements, the system architecture must be designed to make applications readily changeable. There is no point in specifying every system function in exact detail, because most functions will undoubtedly change somewhat during the development. Without an architecture that supports changes, substantial development funds will be spent changing functions that have been optimized to the wrong goal.

This situation arises because the using commands believe they know exactly what is needed at the beginning of the development. The development commands believe their job is to sell the user's defined requirements. Thus, the development process does not promulgate any scheme for thinking about adjustments. In fact, the current process discourages thinking about adjustments because success is defined as meeting the specification.

REDUCING DEPENDENCE ON SPECIFICATIONS

The problem of changing specifications can be addressed by creating a new process, beginning with a team whose only job is to devise an architecture that will support making the system changeable. With a sound architectural basis, even if the team makes many errors building the functions, it will have created a scheme that makes it inexpensive to make changes. Today, most of the effort goes into looking at the user displays and the functions to be sure they are correct, rather than looking at whether or not the architecture is adjustable.

So far as I am aware, the government has never initiated a C^3I development effort by asking for a system architecture explicitly designed for changeability, and it has never asked that a given architecture be evaluated for its ability to support change. No method exists for generating such a specification, and system engineers have no satisfactory way to evaluate architectures. Our community must invent new techniques in this area. We should first recognize that the development of a C^3I system is aimed at a changeable architecture and a first implementation of function -- and then invest vigorous effort in the design of the architecture and perhaps equal effort in the subsequent functional design (but not much more effort in the functional design, as is done today).

A parallel may be drawn with the housing industry. Building a house involves a very mature technology, with two major aspects: (1) the finish-

ing of the house interior, concerning details such as the number of baths and the dimensions of the kitchen, and (2) the basic architectural design of the house.

These two aspects of house-building, finishing and architecture, are handled by two different sets of professionals. Inspectors are employed to make certain that state and local building codes are satisfied. Even though the consumer may not have technical knowledge about the spacing of studs or thickness of concrete, he or she knows it is prudent to have an expert thoroughly check these and other building parameters. Few home-buyers would purchase a house that did not have a careful review by a building codes expert.

With confidence that the house's basic architecture is sound and therefore extendable and changeable, the consumer can specify the details of internal functions with the knowledge that those functions are likely to change as economic circumstances change; for example, the consumer may someday want bigger bathrooms or a larger kitchen.

In software systems, by analogy, the detailed functions inside the house are emphasized while the foundation of the house is virtually ignored. Although the users are fully knowledgeable about the desired functions of the software system, they have little skill in the "building codes" for software. Unfortunately, unlike the housing consumer, the software user as yet has no orientation toward bringing in the equivalent of the building codes inspector who can insist that the basic software architecture be sound and readily accommodate change in the detailed functions.

Performance specifications that focus on completing an architecture create a negative environment for building systems from commercial products, and a negative environment for looking at the architecture from the viewpoint of changeability. How might complex C³I software systems be developed from commercial products, using a process that puts more emphasis on software architecture and is not based on performance specifications? One approach, described below, is to form a team consisting of the users, developers, and industry in various combinations, depending on the development phase.

Understand the User's Needs

The team, which could consist of a mixture of users and developers, must understand the user's desires in detail.

Understand COTS Products

The team must have detailed knowledge of as many available commercial products as possible. There are countless commercial products on the market, and new ones are appearing at a tremendous rate. A substantial effort is needed to understand available products.

It is not practical to set up a special team that would become expert in all the commercial products because the team would be overwhelmed by the volume of products. The only feasible method is to form one or more teams that work on and oversee many different implementations of C³I systems. In that way, knowledge would be gained about not only what products are available, but also how well they integrate and what problems occur with particular integrations. This knowledge would not be comprehensive, but over time would span a large fraction of the necessary ingredients in C³I design.

Learn How to Evaluate System Architecture

Satisfactory methods for evaluating architectures must be developed. Two relatively crude approaches are occasionally employed now. The first involves building prototypes. According to my experience, the general interest in prototypes tends to be oriented to display, analogous to selecting the kind of kitchen the consumer wants in his or her house. The second method involves evaluating prototypes with respect to their architectures. The second method is as crucial to the user, but is not as apparent. As a result, there are few funded efforts devoted to evaluating the architectures of prototypes. Unfortunately, funding that could be used for

this purpose is often expended on fixing systems that have not been properly developed.

An important aspect of architecture evaluation is a method for rejecting products that seem attractive but cannot be extended or expanded within the chosen architecture. There is a strong tendency to incorporate a product if it appears to offer some attractive new feature, without careful examination to see whether the product will allow functional changes at a later date, and without an estimate of the cost to add on the next new commercial capability. Although prototypes can help in estimating the cost of future changes, they are seldom used in this fashion, even though the cost issue is crucial.

COTS-based C³I developments tend to be low-cost projects, driven by time and budget. If the development cost and the user's budget were known at the outset, an informed judgment could be made about what functional capabilities are affordable (whereas a specification must be generated at a time when the feasibility of using COTS products is in doubt, as well as the practicality of future additions). A better process would be to build a prototype capability, learn about cost and architecture from the prototyping exercise, discuss functional performance options and their associated cost with the users, and then, given a budget, decide what is possible.

Bring in Industry Differently

When the time arrives to bring industry into the team, a new method must be found to make the selection from among the bidders for a COTS C³I development. Often, the company selected by the government is the least expensive and the most readily available, because both funds and schedule are severely constrained. With reference to the house-building analogy, it is clear that we would not want to choose the contractor for our home solely on the basis of cost and availability.

Having chosen the basic COTS elements of the architecture, a rational approach is to favor companies that know the most about that class of COTS products, because it is reasonable to expect those companies to be more successful at integrating the products at the lowest cost. In a given development, if one company were found to be most knowledgeable about all of the applicable COTS products, then it would seem appropriate to select that company. On the other hand, following this reasoning, if the selected architecture called for the integration of COTS elements A, B, C, and D, and if company 1 was found to be expert in integrating items A and B while company 2 was expert in integrating items C and D, the government might choose *both* contractors. Unfortunately, the government would be afraid to take this latter course, because in effect the government would be acting as the prime contractor responsible for integrating the system and would therefore be accountable for successful completion of the overall development.

However, the problem of government accountability may not be severe. The government's attitude toward accountability stems from custom-development projects that typically have very high cost (tens to hundreds of millions of dollars); the fear of becoming entangled in expensive litigation is understandable in such cases. By contrast, the scale of COTS C³I developments is typically in the range of four or five million dollars. When the government team has skills and knowledge to do the integration job more professionally than an industrial team (who may not know the user and product selection as well), the government should strongly consider taking on the integrator role.

RELATION OF PRESENT METHODS TO INDUSTRY

While industry might not agree, I believe our current practices in the development of C³I COTS-based systems are harmful to industry. A typical scenario illustrates the point when the development command requests proposals for a C³I system built primarily from COTS products. This request is made even though there is no proof that the specification can be met with COTS products. (Occasionally, feasibility is asserted by the developer because one COTS version of the desired system has been prototyped. However, the selected contractor is free

to choose any set of COTS products. The developer does not know whether the *contractor's* selection of products can do the job, and neither does the contractor, for the reason stated earlier. No contractor can afford to invest in proving feasibility before the bid is made.) After the selection, the contractor begins the development, with the understanding that the specification must be met. When the contractor is unable to meet the specification within schedule, the contractor begins to lose money. Because these jobs are tightly budgeted, it takes only a few months of extra time before the contractor's profits vanish. The results of extended court contests are long delays, increased costs, and undesirable products. Industry bears the risk of meeting the user's requirements with its own selection of off-the-shelf products.

HIGH-RISK AND LOW-RISK PHASES OF DEVELOPMENT

Once the trade-offs of meeting user desires versus the performance of COTS products have been made and are thoroughly understood, the development of COTS-based C^{3}I systems becomes low risk. The trade-offs allow accurate estimation of integration costs and of the cost to incrementally add capability to the system. When the user's real budget is known, and when the user is willing to temper his or her requirements to stay within that budget, the risk becomes small. Generally, very mature COTS software is used so that the risk of encountering severe software bugs is relatively low.

However, before the trade-offs have been completed, the development risk is very high because the contractor is expected to meet a rigid set of requirements but is not allowed (or cannot afford) to do custom design. If the government were to select the products and take the risk that the correct match to the user's needs has been made, then industry could implement the design without being accountable for the match to user requirements, and the risk to industry would be considerably less. Industry should take responsibility for integrating the COTS products well, and should not be concerned about whether the product selection is optimum for the user. The government should be responsible for ensuring the product selection matches user needs.

A NEW DEVELOPMENT APPROACH FOR COTS-BASED C^{3}I SYSTEMS

The preceding discussion results in a different *approach to the development of COTS-based C^{3}I* systems:

- The development command takes responsibility for ensuring that three critical aspects of the development are properly traded off with respect to cost and performance:
  - Balance of the cost to build and modify
  - Availability of products
  - Desires of the user
- The development command takes responsibility for deciding whether, in a given case, it is better to integrate a system with a single contractor or multiple contractors.
- Even if a single contractor is selected, the user and development commands take responsibility for correct product selection. Contractor selection is not based on specifications, but on the contractor's ability to integrate. Under this new procedure, if the performance of the integrated system is poor, then the assessment of blame depends on the cause. If both the contractor and the integration plan prove to be sound, then the development command is at fault. If, however, the integration was not performed well, then the contractor is at fault and should be replaced.
- The development command must have a reasonably comprehensive knowledge of the COTS marketplace to ensure that the final selection of products is chosen from the very large number of commercial products available. The process requires confidence that a good source has not been overlooked, or that some different COTS vendor will not later confront the developer with a superior product.

- The development command must be prepared to evaluate an architecture for basic soundness and the ability for change at reasonable cost. The user must be convinced that the selected architecture is resilient and adaptable. The government's job is to select the system architecture that satisfies the user's requirements at a time when the final system requirements are not yet fully known.
- Finally, the development command must find a way to judge the integration activities performed by the contractor. Today, this issue is confused by the need to meet specifications. When the contractor has conducted a high quality integration effort, it is not appropriate to hold the contractor responsible for the performance of commercial equipment and software.

This summarizes a new model of how COTS-based developments should be done. It is important that an effective approach for this class of development be put in place, because the economics of system development will greatly emphasize the integration of COTS products.

LIFE-CYCLE ISSUES

We are entering a period in which the DOD will not have enough money to replace C³I systems; instead, the systems will have to be maintained and upgraded over a period of many years. It is not clear who will be given this maintenance role — industry, the user, the developer, or the logistics organizations. The issue of organizational role is important and should not be dismissed on the basis of a given command's mission or the assertion that the user must retain control of the process.

If a system composed of COTS products is modified or evolutionary improvements are made, the integrated off-the-shelf products of the system should not be modified because the cost will be high. Any additions should be more off-the-shelf products, and perhaps a small number of custom-designed modules.

The most important factor in upgrading a system should be knowledge of the system's architecture. If a system's architecture has been properly designed to accommodate change, the people who perform the life-cycle maintenance and upgrade role should be experts in that architecture so they can properly exploit the opportunities for change in the system. At present, this assignment of experts does not take place, and no effort is expended to provide the necessary architectural information to the actual maintainers and upgraders. When industry is given the task, the people who do the maintenance are not the designers of the architecture, but rather those who wrote the final application software.

The life cycle is something that we are not thinking about properly, and the issues of transfers of command, transfers of responsibilities, and transfers of money are quite complex. However, this is an area where we must create a more efficient way of doing business. Ideas for handling the full life-cycle of COTS-based C³I systems need development to exploit the opportunities to lower risk and save money.